Arduino dla mikrokontrolerów STM32 (4)

W poprzednich artykułach z tego cyklu pokazano, jak skonfigurować środowisko programistyczne Arduino do pracy z mikrokontrolerami STM32 oraz jak zrealizować aplikację używającą portów wejścia/wyjścia i interfejsu szeregowego UART. Ta część kontynuuje wątek aplikacyjny, demonstrując sposób wykorzystania kolejnego zasobu: przetwornika A/C.

Pracę należy rozpocząć, uruchamiając środowisko programistyczne Arduino. Przygotowanie do pisania kodu obejmuje kilka prostych kroków.

Arduino i STM32 – przygotowanie do pracy

W pierwszej kolejności konieczne jest stworzenie nowego projektu (zwanego sketchem w Arduino). W tym celu użytkownik wybiera z menu głównego środowiska programistycznego kolejno *File* i *New* (skrót klawiaturowy CTRL+N).

W następnym kroku należy wskazać używaną platformę sprzętową. Przykładowo, autor wybrał płytkę NUCLEO-L476RG z 64-pinowym układem STM32L476RG (**rysunek 1**). W tym celu należy ponownie użyć menu głównego środowiska programistycznego. Wybór platformy przebiega w dwóch etapach. Najpierw klikając *Tools* i *Board*, wybrać należy *Nucleo-64*. Następnie *Tools* i *Board Part Number*, wybrać należy *Nucleo L476*.

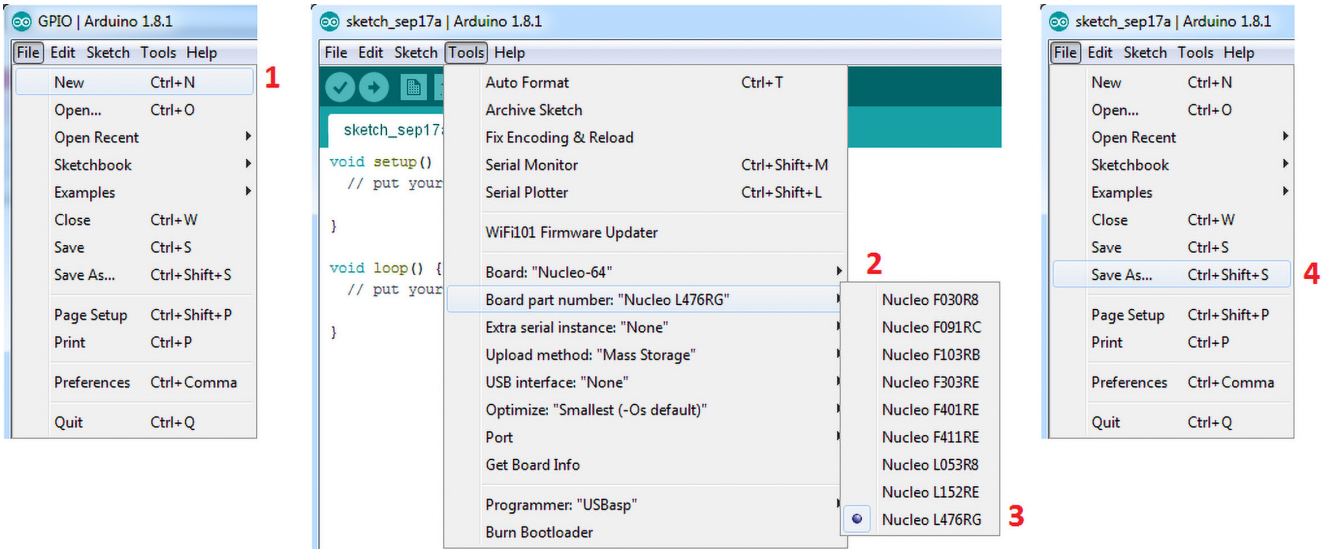
W ostatnim kroku należy zapisać projekt na dysku twardym. Jednocześnie warto zmienić nazwę projektu na bardziej intuicyjną, gdyż domyślna nazwa zawiera tylko słowo sketch oraz nazwę aktualnego miesiąca i dnia. Chcąc zapisać projekt oraz zmienić jego nazwę użytkownik znowu korzysta z menu głównego środowiska programistycznego, wybierając kolejno *File* i *Save as...* (skrót klawiaturowy CTRL+Shift+S). W ten sposób otworzone zostanie okno, które pozwoli na wybranie na dysku twardym ścieżki, pod którą zapisany zostanie projekt. Jednocześnie okno pozwala na wpisanie nowej nazwy projektu.

Arduino i STM32 – struktura kodu

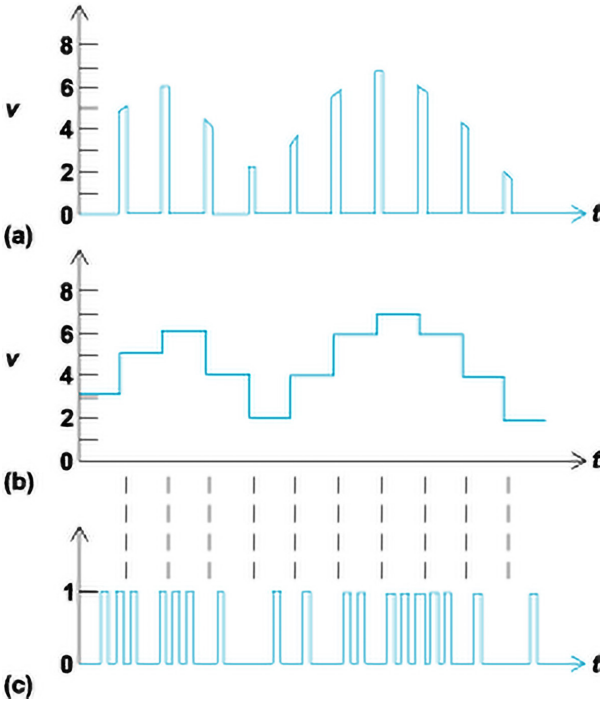
Każdy stworzony od nowa projekt (sketch) zawiera domyślnie kilka linii kodu tworzących szkielet aplikacji. Kod ten pokazano w **listingu 1**. Składa się on z dwóch funkcji. Pierwsza funkcja nosi nazwę *setup*. Jej przeznaczenie można łatwo rozszyfrować, zapoznając się z komentarzem. Programista umieszcza w niej kod, który ma zostać wykonany tylko raz, na początku aplikacji, a więc po doprowadzeniu napięcia zasilania do mikrokontrolera. Są to głównie funkcje konfigurujące peryferia mikrokontrolera (np. ustawiające parametry przetwornika

Listing 1. Kod nowego projektu w środowisku programistycznym Arduino

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```



Rysunek 1. Kolejne kroki przygotowania środowiska Arduino do pisania kodu dla wybranej platformy z układem STM32



Rysunek 2. Kolejne etapy konwersji analogowo-cyfrowej: próbkowanie (a), kwantyzacja (b) i kodowanie (c)

A/C lub interfejsu UART) albo inicjujące podzespoły zewnętrzne (np. moduł Bluetooth). Druga funkcja otrzymała nazwę *loop*. Ponownie w rozszyfrowaniu jej przeznaczenia pomaga komentarz. Jest to nieskończona pętla, w której programista umieszcza funkcje wykonywane cyklicznie (np. odczyt stanu przycisku lub odebranie danych z czujnika). Zatem programista tworzy kod aplikacji, uzupełniając ciała funkcji *setup* i *loop* o dodatkowy kod języka Arduino. Kod ten składa się z typowych elementów języka programowania, takich jak zmienne, funkcje, instrukcje warunkowe czy operacje matematyczne.

Przetwornik A/C – podstawowe informacje

Często zdarza się, że system elektroniczny oparty na mikrokontrolerze wykorzystuje nie tylko sygnały cyfrowe, a więc takie, w których wyróżnić można wartość binarną 0 i 1 (np. sygnał PWM, interfejsy komunikacyjne I²C/UART/SPI itp.), ale używa również sygnałów analogowych, których wartość nie ogranicza się do logicznego 0 i 1, a zmienia się w sposób dowolny w zakresie między wartością minimalną i maksymalną. Do poprawnego odczytania wartości takich sygnałów mikrokontroler wykorzystuje odpowiedni zasób, jakim jest przetwornik A/C. Jak sama nazwa wskazuje, zdefiniować go można jako blok konwertujący sygnał analogowy do postaci cyfrowej. Proces ten odbywa się w trzech etapach: próbkowania, kwantowania i kodowania. Próbkowanie to czynność zamiany sygnału ciągłego na sygnał dyskretny, a więc na grupę pomiarów wykonanych w stałym odstępie czasu. Kwantyzacja jest czynnością polegającą na przypisaniu każdej z próbek wartości liczbowej ze zbioru wartości, jakie przyjąć może wyjście przetwornika. Kodowanie natomiast jest zamianą wartości liczbowej wyjścia przetwornika na odpowiadającą jej sekwencję bitów. Poszczególne etapy tego procesu zilustrowano na **rysunku 2**.

Przetwornik A/C – funkcje Arduino

W Arduino przewidziało wykorzystanie tylko jednego przetwornika A/C. Nie oznacza to jednak, że napięcie można odczytywać tylko z jednego pinu mikrokontrolera. Możliwych do użycia wyprowadzeń mikrokontrolera na potrzeby konwersji analogowo-cyfrowej jest łącznie sześć (oznaczone są one jako A0…A5). Są to tak zwane kanały przetwornika A/C. Wewnątrz mikrokontrolera sygnały z kanałów doprowadzone są do multipleksera, a ten wybiera jeden z nich i łączy go z przetwornikiem.

Sam przetwornik A/C zwraca wynik o rozdzielczości 10 bitów. W praktyce oznacza to, że napięcie minimalne (0 V) reprezentowane jest przez wartość przetwornika 0, natomiast napięcie maksymalne (równe napięciu zasilania, w przypadku płytki Nucleo jest to 3,3 V) reprezentowane jest przez wartość 2 do potęgi 10 minus 1, a więc 1023. Z tej zależności wyznaczyć można wzór przeliczający wynik przetwornika na napięcie

$$Napięcie\ [V] = \frac{Wartość\ przetwornika * 3{,}3[V]}{1023}$$

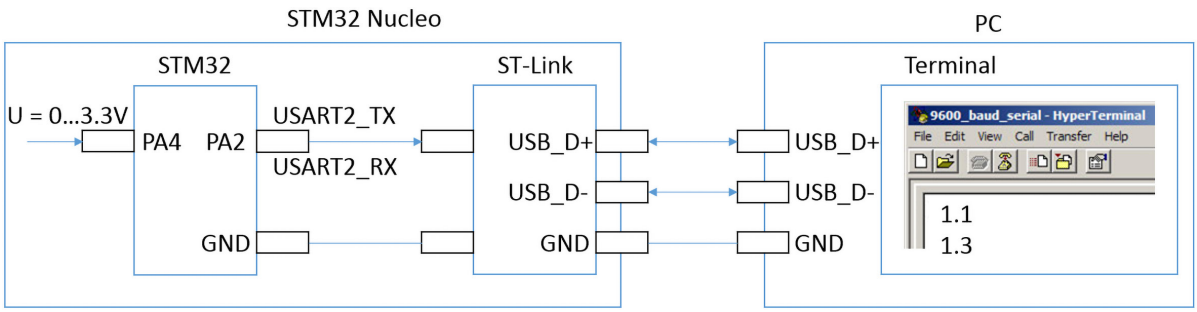
Do korzystania z przetwornika A/C wystarczy znajomość jednej funkcji. Nazywa się ona *analogRead*. Funkcja ta przyjmuje jeden argument, jakim jest liczba całkowita z zakresu 0…5. Liczba ta identyfikuje wejście przetwornika (kanał), którego programista chce użyć, a więc A0…A5. Funkcja zwraca wartość adekwatną do odczytanego napięcia (zakres 0…1023).

Przetwornik A/C – przykładowa aplikacja

W odstępnie czasu co 100 ms aplikacja będzie odczytywała wynik konwersji przetwornika A/C. W przykładowej aplikacji wykorzystany zostanie jeden kanał przetwornika oznaczony jako A3 (pin PA4). Za każdym razem odczytana wartość zostanie przekształcona za pomocą wzoru na wartość napięcia. Następnie wynik napięcia zostanie wysłany portem szeregowym (pin PA2). Jako że zintegrowany na płytce Nucleo programator/debugger ST-Link pełni funkcję mostu pomiędzy interfejsem UART i interfejsem USB emulującym port COM, wysyłane przez aplikację wartości będzie można odczytać za pomocą terminalu komputerowego. Odpowiedni schemat pokazano na **rysunku 3**.

Po zapoznaniu się najpierw z funkcją Arduino do obsługi przetwornika A/C, a następnie po przeanalizowaniu schematu połączeń, jesteśmy gotowi do napisania programu realizującego odczyt wartości przetwornika, konwersję do wartości napięcia oraz transmisję danych do komputera. Zaprezentowany na **listingu 2** kod zawiera kolejno:

- Przed funkcją *setup*:
  - deklarację zmiennej typu całkowitego *Wartosc* do przechowywania wyniku konwersji przetwornika A/C,
  - deklarację zmiennej typu zmiennoprzecinkowego *Napiecie* do przechowywania wyniku napięcia,



Rysunek 3. Schemat połączeń na potrzeby aplikacji

- wewnątrz funkcji *setup*:
  - wywołanie funkcji *Serial.begin* ustawiającej prędkość transmisji interfejsu UART,
- wewnątrz funkcji *loop*:
  - wywołanie funkcji *analogRead* realizującej pomiar przetwornikiem A/C dla wybranego kanału i zwracającej wynik tej konwersji, która zapisana zostaje w zmiennej *Wartosc*,
  - przekształcenie za pomocą wzoru wyniku konwersji na napięcie i zapisanie jej w zmiennej *Napiecie*,
  - wywołanie funkcji *Serial.println*, która wysyła do komputera zawartość zmiennej *Napiecie*.

Po napisaniu kodu program należy skompilować, wybierając przycisk *Verify* (skrót klawiaturowy CTRL+R). Po zakończeniu procesu kompilacji należy wgrać program do pamięci mikrokontrolera, wybierając przycisk *Upload* (skrót klawiaturowy CTRL+U).

Działanie aplikacji można obserwować z poziomu środowiska programistycznego Arduino IDE, które dysponuje terminalem komputerowym portu COM oraz programem do wizualizacji danych. Terminal można włączyć z poziomu menu głównego, wybierając kolejno *Tools* i *Serial Monitor* (skrót klawiaturowy CTRL+Shift+M). Program do wizualizacji danych również jest dostępny z poziomu menu głównego: *Tools* i *Serial Plotter* (skrót klawiaturowy CTRL+Shift+L). Widok okien obu programów podczas działania aplikacji mikrokontrolera pokazano na **rysunku 4**.

Listing 2. Kod programu realizującego odczyt wartości przetwornika A/C, konwersję do wartości napięcia oraz transmisję danych do komputera

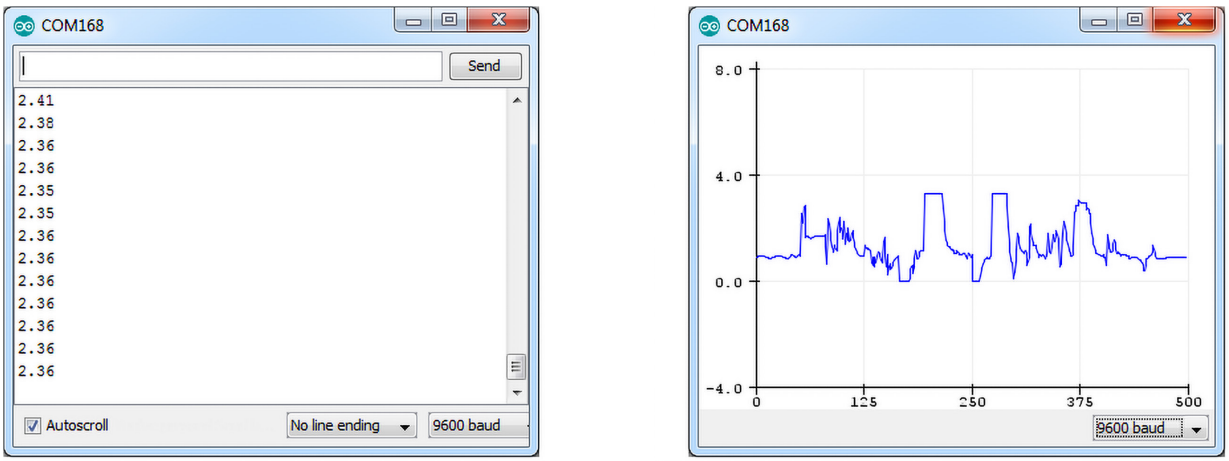
```
int Wartosc = 0;
float Napiecie = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop()
{
  // put your main code here, to run repeatedly:
  Wartosc = analogRead(3);
  Napiecie = Wartosc * (3.3/1023);
  Serial.println(Napiecie);
  delay(100);
}
```

Podsumowanie

Czwarty artykuł z cyklu traktującego o Arduino i STM32 kontynuuje wątek aplikacyjny, koncentrując się tym razem na przetworniku A/C. W przedstawionym materiale pokazano, jak realizować pomiar napięcia przetwornikiem oraz zademonstrowano sposób wysłania tych danych do komputera za pomocą interfejsu szeregowego.

Kolejny artykuł zakończy aspekt prostych aplikacji korzystających z zasobów mikrokontrolera. Tematyką będzie timer i wytwarzanie za jego pomocą sygnału PWM.

Szymon Panecki
STMicroelectronics
szymon.panecki@st.com



Rysunek 4. Widok programów do obsługi komunikacji szeregowej w Arduino IDE. Od lewej: terminal, program do wizualizacji danych

REKLAMA

MEDIA ELEKTRONIKA PRAKTYCZNA

Od stycznia br. zmieniliśmy sposób dostarczania Czytelnikom EP materiałów dodatkowych dołączonych do numeru.

1. Wejdź na stronę **www.media.avt.pl**
2. Zarejestruj się/zaloguj
3. Wybierz wydanie „Elektroniki Praktycznej", które chcesz dodać do swojej biblioteki.
4. Odpowiedz na proste pytanie dotyczące bieżącego numeru.
5. Pobieraj pliki.